CLAIMS

1. (Currently Amended) A computer implemented method of analyzing a multi-threaded program[[s]], comprising:

suspending a first thread , the first thread holding a first synchronized object and requesting ~~that requests~~ a second synchronization object that could result in a deadlock if acquired;

receiving a request from a second thread to acquire the second synchronization object while the first thread is suspended;

allowing the second thread to acquire the second synchronization object; and

causing a latent deadlock to manifest by awakening the first thread ~~to awake~~ in response to an event message, the awakened first thread resuming requesting the second synchronization object while the second ~~to potentially produce a deadlock;~~

~~wherein the method of suspending the first thread is in response to the first thread's request for the synchronization object that could result in a deadlock if acquired so evidenced by existence if any~~ thread is currently holding but not having released the second synchronization object, the latent deadlock manifesting itself when the second thread or a third thread attempts to acquire the first ~~while acquiring another~~ synchronization object; and

detecting the latent deadlock including indicating cause of the latent deadlock in a manner enabling a user to remedy problems of the multi-threaded program.

2. (Currently amended) The method of claim 1, further comprising checking whether the first and second thread are deadlocked by the first thread waiting to acquire [[a]] the second synchronization object that the second thread holds and the second thread waiting to acquire [[a]] the first synchronization object that the first thread holds.

3. (Previously presented) The method of claim 1, wherein the first thread is suspended for a predetermined time.

4. (Previously presented) The method of claim 3, wherein the first thread is also suspended on an event.

5. (Original) The method of claim 4, wherein the second thread sends the event that awakens the first thread.

6. (Original) The method of claim 1, wherein the first and second threads can hold a plurality of synchronization objects at a time.

7. (Currently amended) The method of claim 1, wherein only one thread can hold the second synchronization object at a time.

8. (Original) The method of claim 1, wherein only the first and second threads can release synchronization objects that each holds.

9. (Currently Amended) A computer program product for analyzing multi-threaded programs, comprising:

a computer ~~useable~~ storage medium to memory having a computer readable program, wherein the computer readable program when executed on a computer causes the computer to:

~~request to~~ track a first thread, the first thread holding a first synchronization object and requesting a second synchronization object that could result in a deadlock if acquired;

determine evidence of such ~~resulting~~ deadlock;

suspend the first thread;

receive a request from a second thread to acquire the second synchronization object while the first thread is suspended;

allow the second thread to acquire the second synchronization object; and

cause a latent deadlock to manifest by awakening the first thread to awake in response to an event message, the awakened first thread resuming requesting the second synchronization object while the second ~~to potentially produce a deadlock;~~

~~wherein the computer readable program suspend the first thread is in response to the first thread's request for the synchronization object that could result in a deadlock if acquired so evidenced by existence if any~~ thread is currently holding but not having released the second synchronization object, the latent deadlock manifesting itself when the second thread or a third thread attempts to acquire ~~while acquiring another~~ the first synchronization object; and

detecting the latent deadlock including indicating cause of the latent deadlock in a manner enabling a user to remedy problems of the multi-threaded program.

10. (Currently Amended) The computer program product of claim 9, wherein the computer ~~useable~~ storage medium to memory is any of a CD-ROM, floppy disk, tape, flash memory, system memory, and a hard drive.

11. (Currently Amended) The computer program product of claim 9, wherein the computer readable program checks whether the first and second thread are deadlocked by the first thread waiting to acquire [[a]] the second synchronization object that the second thread holds and the second thread waiting to acquire [[a]] the first synchronization object that the first thread holds.

12. (Previously presented) The computer program product of claim 9, wherein the first thread is suspended or a predetermined time.

13. (Previously presented) The computer program product of claim 12, wherein the first thread is also suspended on an event.

14. (Previously presented) The computer program product of claim 13, wherein the second thread sends the event that awakens the first thread.

15. (Previously presented) The computer program product of claim 9, wherein the first and second threads can hold a plurality of synchronization objects at a time.

16. (Currently amended) The computer program product of claim 9, wherein only one thread can hold the second synchronization object at a time.

17. (Previously presented) The computer program product of claim 9, wherein only the first and second threads can release synchronization objects that each holds.

18. (Previously presented) The computer program product of claim 9, further comprising causing the first thread to awake in response to an expiration of time.

19. (Previously presented) The method of claim 1, further comprising causing the first thread to awake in response to an expiration of time.